senseable city lab:.::

A General Optimization Technique for High Quality Community Detection in Complex Networks

Stanislav Sobolevsky *† Riccardo Campari†
Alexander Belyi‡ Carlo Ratti†

August 19, 2013

Abstract

Recent years have witnessed the development of a large body of algorithms for community detection in complex networks.

Most of them are based upon the optimization of objective functions, among which modularity is the most common, though a number of alternatives have been suggested in the scientific literature.

We present here an effective general search strategy for the optimization of various objective functions for community detection purposes. When applied to modularity, on both real-world and synthetic networks, our search strategy substantially outperforms the best existing algorithms in terms of final scores of the objective function; for description length, its performance is on par with the original Infomap algorithm.

The execution time of our algorithm is on par with non-greedy alternatives present in literature, and networks of up to 10,000 nodes can be analyzed in time spans ranging from minutes to a few hours on average workstations, making our approach readily applicable to tasks which require the quality of partitioning to be as high as possible, and are not limited by strict time constraints.

Finally, based on the most effective of the available optimization techniques, we compare the performance of modularity and code length as objective functions, in terms of the quality of the partitions one can achieve by optimizing them. To this end, we evaluated the ability of each objective function to reconstruct

*To whom correspondence should be addressed. E-mail: stanly@mit.edu

†SENSEable City Laboratory,Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA,

‡Institute of Mathematics, National Academy of Sciences of Belarus Republic, 11 Surganova str., Minsk, Belarus

the underlying structure of a large set of synthetic and real-world networks.

Complex networks — Community detection — Network science

The increasing availability of big data has motivated an enormous general interest in the burgeoning field of network science.

In particular, the broad penetration of digital technologies in different spheres of human life provides substantial sources of data sets which explore the intricacies of manifold aspects of human activity. The topics they cover range from personal relationships among individuals to professional collaborations, from telephone communication to data exchange, from mobility and transportation to economical transactions and interactions in social media. Analyzing such data sets often leads to the construction of complex networks describing relations among individuals, enterprises, locations, or more abstract entities, such as the buzzwords and hashtags employed in social media; whenever the resulting structures are geographically located, they can then be studied at different scales, including global, countrywide, regional, and local levels. Furthermore, complex networks can arise from the study of biological phenomena, including neural, metabolic, and genetic interactions.

Community detection is one of the pivotal tools for understanding the underlying structure of complex networks and extracting useful information from them; it has been used in fields as diverse as biology [1], economics - the World Trade Net is analyzed in [2] - human mobility [3–5], and scientific collaborations [6].

Many algorithms were devised in the field of community detection, ranging from straightforward partitioning approaches, such as hierarchical clustering [7] or the Girvan-Newman [8] algorithm, to more sophisticated optimization techniques based on the maximization of various objective functions.

The most widely used objective function for partitioning is modularity [9,10]: it relies on comparing the strength of inter- and intra-community connections with a null-model in which edges are randomly re-wired.

In order to obtain partitions yielding optimal values for modularity, researchers have suggested a large number of optimization strategies: well-known algorithms include the simple greedy agglomerative optimization by Newman [11] and faster Clauset-Newman-Moore heuristic [12], Newman's spectral division method [9] and its improvements (which employ an additional Kernigan-Lin-style [13] step), [10], the aggregation technique commonly referred to as Louvain method, extremely fast even on large-scale networks [14], simulated annealing [15, 16], extremal optimization [17] and many others [18].

In the last few years, researchers have shown that modularity suffers from certain drawbacks, including a resolution limit [16, 19] which prevents it from recognizing smaller communities (a proposed multi-scale workaround which involves modifying the network can be found in [20]).

At least three of the several alternative objective functions deserve to be mentioned: description length, block model likelihood measure, and surprise.

The description length of a random walk on a network, upon which the Infomap algorithm [21, 22] by Rosvall and Bergstrom is based, is an well-known information-theoretical measure, reputed to be among the best available [23]; it appears, however, that code length optimization also suffers from a resolution limit, as discussed in [24], where a workaround is proposed.

The second approach is based on the likelihood measure for the stochastic block model suggested in [25–27].

Finally, Surprise [28] compares the distribution of inter-community links to that emerging from a random network with the same distribution of nodes per community.

For a detailed, if not up-to-date, review of existing community detection methods, the reader can refer to Ref. [18].

A few more strategies for community detection follow: the replica correlation method introduced in [29], which is also an information-based measure; two recently proposed algorithms, which infer community structures by using generalized Erdős Numbers [30] and by focusing on the statistical significance of communities [31]; a recent approach for modularity optimization - conformational space annealing [32] - which delivers acceptable results very quickly, and is scalable to larger networks, as is the modification to the algorithm by Clauset, Newman, and Moore [33] proposed in [34].

A key point in the evaluation of algorithms for community detection is the choice of meaningful benchmarks. Benchmarks can be roughly divided into two groups.

In the first, one compares the final scores achieved by different algorithms for the optimization of the same objective function on a variety of networks.

In the second type of benchmark, resulting partitions are checked against imposed or well-known structures in synthetic or real-world networks; this kind of benchmark is fundamental for the evaluation of different partitioning techniques not necessarily based on the optimization of the same objective function.

Other methods to obtain independent evaluations of the reliability of communities found, without relying on the known community structure nor objective function scores, focus - among other parameters - on recurrence of communities under random walks [35, 36], and their resilience under perturbations of the network edges [37].

In the present work we suggest a novel universal optimization technique for community detection, which we apply to two of the aforementioned objective functions: modularity and description length.

We also present the results of a two-stages benchmark. First, we compare the performance of our algorithm, in terms of the resulting values for objective functions, with a host of existing optimization strategies, separately for modularity and description length; we show in this way that we consistently provide the best modularity scores, and results on par with Infomap when optimizing description length.

Next, by employing in each case the best available algorithm, we compare the performances of modularity and description length as objective functions

in reconstructing underlying structures on a large set of synthetic networks, as well as the known structures on a set of real-world networks.

1. The algorithm

The vast majority of search strategies take one of the following steps to evolve starting partitions: merging two communities, splitting a community into two, moving nodes between two distinct communities.

The suggested algorithm involves all three possibilities. After selecting an initial partition made of a single community, the following steps are iterated as long as any gain in terms of the objective function score can be obtained: (**1**) for each source community, the best possible redistribution of every source nodes into each destination community (either existing or new) is calculated; this also allows for the possibility that the source community entirely merges with the destination; (**2**) the best merger/split/recombination is performed. As the proposed technique combines all three possible types of steps, in the following we'll refer to it as Combo.

The fulcrum of the algorithm is the choice of the best recombination of vertices between two communities, as splits and mergers are particular cases of this operation: for each pair of source and (possibly empty) destination communities, we perform a shift of all the vertices fashioned after Kernigan and Lin's algorithm [13]. Specifically, first we initialize the list of available nodes, including all the nodes currently in the source community, then we iterate the following steps until no further improvement of the objective function can be obtained: (**a**) find the node i from the list for which switching community entails the largest gain or the lowest loss (if no gains are available); (**b**) switch i to the other community removing i from the list of available nodes and saving the intermediate result. When no further gain is possible, the best intermediate result is selected as the output of the series.

Experimental tests show a striking regularity in the dependence of the execution time of Combo on the number of nodes of the network; as shown in Fig.1, this behaviour is compatible with a power law with exponent 2.

As the sequence of operations in Combo is strongly dependent on the specific network, sharp evaluations of its computational complexity are difficult to obtain; the striking regularity of the dependence observed in Fig.1 - however - hints at some robust mechanism acting under the hood. In the Supplementary Material, we justify an upper bound to the execution time of $\mathcal{O}\left(N^2 \log\left(\mathcal{C}\right)\right)$, where N is the number of nodes, and $\mathcal{C}$ the number of communities in the network.

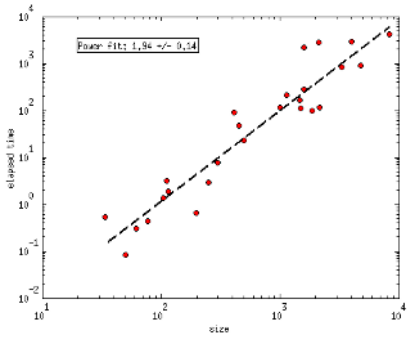

Figure 1: For Combo, the variation of convergence time with the size of the network is compatible with a square power law.

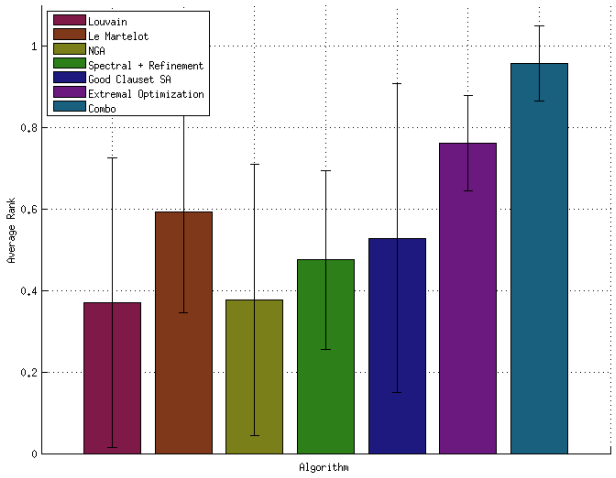

Figure 2: We plot here the average normalized rank per algorithm: values ranging from 0 (worst performance) to 1 (best) are attributed to each algorithm, and their average computed. Standard deviations are also plotted.

2. Modularity optimization benchmarks

We first evaluated the performance of Combo for modularity optimization. We selected six algorithms for the comparison: **a)** Louvain method [14]; **b)** Le Martelot [36]; **c)** Newman's greedy algorithm (NGA) [11]; **d)** Newman's spectral algorithm with refinement [10]; **e)** Simulated annealing [15], in the implementation by Good, Montjoye, and Clauset [16]; **f)** Extremal optimization [17].

The set of algorithms we have chosen offers a good sample of the current state of the art. Simulated annealing is reputed to be capable of getting very close to real maxima, and extremal optimization offers a good tradeoff between speed and performance [18,38,39]; they resulted the best-performing algorithms in at least one benchmark [40]. The recursive Louvain method is fast and relatively effective [23] and has therefore been applied in various real-world network analyses [41, 42]. Newman's greedy algorithm and Spectral Algorithms can be considered classical approaches, since they were suggested right after modularity was introduced about 10 years ago, and were therefore used in a number of previous benchmarks [14, 18, 23, 39]. The technique by Le Martelot is a more recent approach, for which a benchmark already exists [43].

We ran each algorithm on three sets of networks: **(1)** widely available data sets found in literature; **(2)** five graphs - obtained from NDA-protected telecom data - in which the weight of each edge corresponds to the total duration of telephone calls between two locations; **(3)** five synthetic networks generated using the Lancichinetti-Fortunato-Radicchi approach [44, 45]. Detailed descriptions and references can be found in the Supplementary Material.

As a measure of the comparative quality of partitioning, we computed the average rank of each algorithm over all the networks on which it has been tested. When multiple algorithms yielded the same modularity, we equated their rank to the best among them (1 for the highest modularity score).

As summarized in Fig.2, Combo significantly outperforms other algorithms, with an average score of 0.96; the next best placements are Extremal Optimization (0.76), Le Martelot (0.60), and Good and Clauset's Simulated Annealing implementation (0.53); however, the two previous algorithms only work for symmetric matrices. Other algorithms show considerably less consistent outcomes.

The quality improvements obtained often come at the price of execution times, which - as presented in details in the Supplementary Material - show that Combo, which is currently implemented as a Matlab script, is not as fast as the greedy algorithms (Louvain, Spectral), but results on several occasions faster than other algorithms, both complex, such as Simulated Annealing, and simple, as NGA (for which we are however using a Matlab implementation). In the worst cases (usually when the resulting number of communities is big enough), Combo finalizes computation in a matter of hours for the considered networks of the scale of thousands of nodes. Detailed execution times for all the algorithms are reported in the Supplementary Material. It's also noteworthy that a considerable speedup may be obtained by porting Combo to a compiled language.

In cases where the network is big enough, the computational time is crucial, while the resulting partitioning quality is not, using the faster approaches might

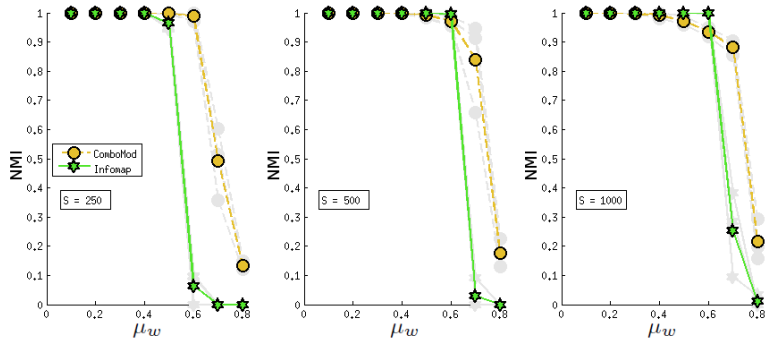

Figure 3: We present here a comparison between optimization of modularity and code length. The x coordinate represents the mixing factor μ_w; the y coordinate is the normalized mutual information. The topological mixing factor μ_t is set to 0.5. Light gray lines are realizations of networks with $15, 20, 25$ neighbours, while their averages are represented by color lines.

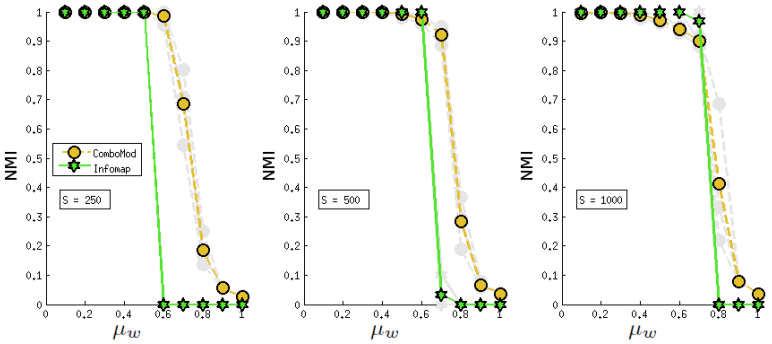

Figure 4: The topological mixing factor μ_t is equal to μ_w.

be the better choice.

Often, however, the reliability of the final community structure is of paramount importance: in such cases, we'll want to aim at the highest possible value of the objective function, as even even small differences in the resulting modularity score can translate into macroscopic variations in the quality of partitioning. In the Supplementary Material, we show that a variation as small as 0.5% can have a sizable impact on the community structure of a network. While at the moment it's impossible to guarantee that an achieved partition is a global maximum, we can assume that choosing the one sporting the highest score is the best option.

3. Minimum description length benchmarks

In our second benchmark, we use the combo algorithm to optimize description

length compression, and compare the results to those obtained using the original Infomap implementation by Rosvall and Bergstrom [21, 22].

We ran the comparison on the same set of networks as in the previous benchmark. Since Infomap is a greedy algorithm and results are dependent on a random seed, we ran it 10 times for each network and picked the best result.

Unlike for modularity, final values for code length are very close, with a single network in which their difference is about 5%, and less than 3% in all other cases; Combo yields a better code length in 8 networks, Infomap in 9, the results being the same in all other cases. Detailed results are reported in the Supplementary Material.

Combo thus results a valid alternative and an ideal complement to Infomap, as in several cases it's proved capable of finding better solutions.

4. Synthetic and Real World Networks Benchmark

After validating that the performance of Combo is optimal for modularity optimization purposes and on par with Infomap for code length, let us use these techniques to compare the performance of modularity and code length as objective functions, i.e. as to how each of them reproduces preimposed structures in random networks; here we generated them following Lancichinetti, Fortunato and Radicchi [44, 45] benchmark approach. Some attempts at comparing multiple partitioning algorithms are already present in literature [46]; here, we specialize the comparison to code length and modularity, using the top-performing algorithms for each; this is a key step, as we wish to compare the efficiency of the objective functions themselves, rather then the performance of each particular optimization technique.

Our implementation of this benchmark consists of two main sets of networks: in the first, we set the mixing parameter for links topology (which governs how many inter-community links are generated), μ_t, to 0.5 (see Supplementary Material); in the second, we chose $\mu_t = \mu_w$, where μ_w is the varying mixing parameter for network weights (likewise, it decides how strong inter-community links are). In each set, we varied the size of the network $(250, 500, 1000)$ and the average degree of the nodes $(15, 20, 25)$. For each chosen set of parameters, we generated ten networks, and on each of them we ran community detection for modularity (via Combo) and description length (via Infomap).

To quantitatively compare resulting communities with the original partition, we employed normalized mutual information (NMI) [40], the definition of which is given in the Supplementary Material.

Results are reported in Figs. 3-4: code length does a slightly better job reconstructing the original communities for lower values of network weight mixing parameter, in particular for higher node counts; on the other hand, its performance drops extremely fast above $\mu_w \simeq 0.5$, while modularity performance decays more slowly, in accordance with similar findings in Ref. [23, 31].

Based on that, one could recommend using modularity for discovering the community structure in networks with weaker clustering effect, while code length might be a better choice for larger networks with relatively strong communi-

Table 1: Comparison between original (Or) communities and those resulting from the optimization of modularity (Mod) and code length (CL) on real-world networks, including number of communities (NC) and Normalized Mutual Information (NMI) (with respect to the original community structure).

Network	Size	NC Or	NC Mod	NC CL	NMI Mod	NMI CL
football	115	12	12	10	0.890317	0.924195
karate	34	2	3	4	0.687263	0.825518
macaque	45	2	3	3	0.639544	0.753089
UKfaculty	81	4	10	5	0.788002	0.660034
polbooks	105	3	6	5	0.560263	0.493454
polblogs1222	1222	2	45	7	0.616725	0.433617

ties. Further benchmarks, conducted for less well-known classes of synthetic networks and described in the Supplementary Material, yield similar results. It is important to note that the scope of these results is limited to this specific types of random network studied.

While real networks seldom have any kind of true or *a priori* structure with which we can compare the quality of community reconstruction, we were able to identify six such networks – thoroughly described in the Supplementary Material – in the scientific literature, and compare their underlying structure with the communities obtained by optimizing modularity and description length.

The results are summarized in table 1: it is apparent that none of the two objective functions is consistently better at reconstructing the known structure of the network. Although it could be argued that modularity performs better in more complex cases when the number of nodes is larger, the number of networks is far too small for reliable generalizations.

We should also stress that, when dealing with real networks, one has to keep in mind that the background communities, defined on the basis of non-structural information, are not necessarily reflected by the actual connections between nodes.

Thus, communities detected by methods based only on the graph structure don't necessary have to coincide with "natural" divisions, as our information measures combine complexity – realized as the mechanism through which the underlying structure is translated into inter-node relationships (which is essentially unknown) – random noise, and individual quirks of the the objective functions.

5. Conclusions

We have presented Combo, an optimization algorithm for community detection capable of handling various objective functions, and we have applied it to the optimization of the two most popular partitioning quality measures: modularity

and description code length. With regard to modularity, Combo consistently outperforms all the other algorithms with which we have compared it, including the current state of the art. For what concerns the optimization of code length, Combo provides results on par with those of Infomap, which is the defining algorithm for this objective function.

Running times of Combo are longer than with greedy algorithms, such as the Louvain method, and on par with more complex ones; often, they are considerably shorter than for extremal optimization or simulated annealing. Even for networks consisting of several thousands of nodes, the algorithm converges in under an hour on consumer-level workstations. Combo is thus an optimal choice when the quality of the resulting community is of paramount importance, and no strict limits are imposed on computation time. Due to memory limitations, the current algorithm implementation is not widely scalable, and its application limit is of the order of ten thousand nodes. Alternative implementations may overcome the current limit.

Combo is also flexible, in that it can be adapted to different objective functions; possible extensions include stochastic block model likelihood [25] and surprise [28]. Additional advantages include the possibility of limiting the number of resulting communities (e.g. to obtain the optimal bi-partitioning of a network) and the applicability of the algorithm to fine-tune the outcomes of other algorithms.

Finally, by studying how well the most efficient optimization techniques for modularity and code length reproduce the underlying community structure of the network, we have provided as fair as possible a comparison between the two objective functions.

On the sample of random graphs generated according to the Lancichinetti-Fortunato-Radicchi approach, description length initially achieves a slightly better fidelity in reconstructing the stronger network structure; however, above a certain threshold value of the weight mixing parameter, resulting partitions quickly deteriorate, while modularity results substantially more resilient to the introduction of noise, in accordance with existing results in literature [23, 31]. We also compared, for the first time, the results of the optimization of modularity and code length for a small set of real world networks with a known underlying structure: although modularity yielded better results for more complex networks, neither emerged as the better approach.

Acknowledgements

Thanks to the National Science Foundation, the AT&T Foundation, the MIT SMART program, the MIT CCES program, Audi Volkswagen, BBVA, The Coca Cola Company, Ericsson, Expo 2015, Ferrovial, GE and all the members of the MIT Senseable City Lab Consortium for supporting the research.

The authors also want to thank Paolo Santi for helpful discussions.

References

[1] Roger Guimera and Luis A. Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, Feb 2005.

[2] Carlo Piccardi and Lucia Tajoli. Existence and significance of communities in the world trade web. *Phys. Rev. E*, 85:066119, Jun 2012.

[3] Christian Thiemann, Fabian Theis, Daniel Grady, Rafael Brune, and Dirk Brockmann. The structure of borders in a small world. *PLoS ONE*, 5(11):e15422, 11 2010.

[4] T. Hossmann, T. Spyropoulos, and F. Legendre. A complex network analysis of human mobility. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 876 –881, april 2011.

[5] Carlo Ratti, Stanislav Sobolevsky, Francesco Calabrese, Clio Andris, Jonathan Reades, Mauro Martino, Rob Claxton, and Steven H. Strogatz. Redrawing the map of great britain from a network of human interactions. *PLoS ONE*, 5(12):e14248, 12 2010.

[6] Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, Jun 2005.

[7] Trevor Hastie. *The elements of statistical learning : data mining, inference, and prediction : with 200 full-color illustrations*. Springer, New York, 2001.

[8] M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99 (12):7821–7826, 2002.

[9] M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69 (2):026113, 2004.

[10] M.E.J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[11] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, Jun 2004.

[12] A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev.*, E70 (6):066111, 2004.

[13] B. W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell system technical journal*, 49(1):291–307, 1970.

[14] V.D. Blondel, J.L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech*, 10008, 2008.

[15] L.A.N. Amaral R. Guimerà, M. Sales-Pardo. Modularity from fluctuations in random graphs and complex networks. *Phys, Rev.*, E70(2):025101, 2004.

[16] Benjamin H. Good, Yves-Alexandre de Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Phys. Rev. E*, 81:046106, Apr 2010.

[17] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Phys. Rev. E*, 72:027104, Aug 2005.

[18] S. Fortunato. Community detection in graphs. *Physics Report*, 486:75–174, 2010.

[19] Santo Fortunato and Marc Barthlemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[20] A Arenas, A Fernndez, and S Gmez. Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics*, 10(5):053039, 2008.

[21] Martin Rosvall and Carl T. Bergstrom. An information-theoretic framework for resolving community structure in complex networks. *Proceedings of the National Academy of Sciences*, 104(18):7327–7331, 2007.

[22] M. Rosvall and C.T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proc. Natl. Acad. Sci. USA*, 105:1118–1123, 2008.

[23] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: A comparative analysis. *Phys. Rev. E*, 80:056117, Nov 2009.

[24] L. Karl Branting. Information theoretic criteria for community detection. In *Proceedings of the Second international conference on Advances in social network mining and analysis*, SNAKDD'08, pages 114–130, Berlin, Heidelberg, 2010. Springer-Verlag.

[25] Peter J. Bickel and Aiyou Chen. A nonparametric view of network models and newmangirvan and other modularities. *Proceedings of the National Academy of Sciences*, 2009.

[26] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Inference and phase transitions in the detection of modules in sparse networks. *Phys. Rev. Lett.*, 107:065701, Aug 2011.

[27] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Phys. Rev. E*, 84:066106, Dec 2011.

[28] Rodrigo Aldecoa and Ignacio Marn. Deciphering network community structure by surprise. *PLoS ONE*, 6(9):e24195, 09 2011.

[29] Peter Ronhovde and Zohar Nussinov. Multiresolution community detection for megascale networks by information-based replica correlations. *Phys. Rev. E*, 80:016109, Jul 2009.

[30] Greg Morrison and L. Mahadevan. Discovering communities through friendship. *PLoS ONE*, 7(7):e38704, 07 2012.

[31] Andrea Lancichinetti, Filippo Radicchi, Jos J. Ramasco, and Santo Fortunato. Finding statistically significant communities in networks. *PLoS ONE*, 6(4):e18961, 04 2011.

[32] Juyong Lee, Steven P. Gross, and Jooyoung Lee. Modularity optimization by conformational space annealing. *Phys. Rev. E*, 85:056702, May 2012.

[33] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111, Dec 2004.

[34] Ken Wakita and Toshiyuki Tsurumi. Finding community structure in mega-scale social networks: [extended abstract]. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 1275–1276, New York, NY, USA, 2007. ACM.

[35] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona. Stability of graph communities across time scales. *Proceedings of the National Academy of Sciences*, 107(29):12755–12760, 2010.

[36] Erwan Le Martelot and Chris Hankin. Multi-scale community detection using stability as optimisation criterion in a greedy algorithm. In *Proceedings of the 2011 International Conference on Knowledge Discovery and Information Retrieval (KDIR 2011)*, pages 216–225, Paris, October 2011. SciTePress.

[37] Atieh Mirshahvalad, Johan Lindholm, Mattias Derln, and Martin Rosvall. Significant communities in large sparse networks. *PLoS ONE*, 7(3):e33721, 03 2012.

[38] Jian Liu and Tingzhan Liu. Detecting community structure in complex networks using simulated annealing with -means algorithms. *Physica A: Statistical Mechanics and its Applications*, 389(11):2300 – 2309, 2010.

[39] Rodrigo Aldecoa and Ignacio Marìn. Surprise maximization reveals the community structure of complex networks. *Sci. Rep.*, 3, Jan 2013.

[40] Leon Danon, Albert Daz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, 2005.

[41] David Meunier, Renaud Lambiotte, Alex Fornito, Karen Ersche, and Edward T Bullmore. Hierarchical modularity in human brain functional networks. *Frontiers in Neuroinformatics*, 3(37), 2009.

[42] Lin Zhang, Xinhai Liu, Frizo Janssens, Liming Liang, and Wolfgang Gl˙ Subject clustering analysis based on {ISI} category classification. *Journal of Informetrics*, 4(2):185 – 193, 2010.

[43] Erwan Le Martelot and Chris Hankin. Multi-scale community detection using stability optimisation within greedy algorithms. *CoRR*, abs/1201.3307, 2012.

[44] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78 (4):046110, 2008.

[45] A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E.*, 80(1):016118, 2009.

[46] Twan van Laarhoven and Elena Marchiori. Graph clustering with local search optimization: The resolution bias of the objective function matters most. *Phys. Rev. E*, 87:012812, Jan 2013.

A General Optimization Technique for High Quality Community Detection in Complex Networks - Supplementary Material

Stanislav Sobolevsky *† Riccardo Campari†
Alexander Belyi‡ Carlo Ratti†

August 19, 2013

1 The Objective Functions

1.1 Modularity

Modularity [1] is probably the best known and most used among objective functions for community detection. It is defined as

$$Q = \sum_{i,j} Q_{ij} \delta\left(C_i, C_j\right), \tag{1}$$

where

$$Q_{ij} = \frac{1}{2m}\left(W_{ij} - \frac{S_i T_j}{2m}\right); \tag{2}$$

i, j are nodes, C_i, C_j the communities they belong to, W_{ij} is the weight matrix, $S_i = \sum_j W_{ij}$, $T_j = \sum_i W_{ij}$, $m = \frac{1}{2}\sum_{ij} W_{ij}$; $\delta(x, y) = 1$ if $x = y$, 0 otherwise.

The idea behind Modularity is to compare the partition to a null model where the network undergoes a node weight-preserving rewiring; modularity scores reflect the simple idea that in good community structures links between nodes of the same community should be generally stronger than null model expectations, while links between different communities should be weaker.

More in detail, the null model is formed by

1. preserving the total out-weight (S_i) and in-weight (T_j) for each node;

*To whom correspondence should be addressed. E-mail: stanly@mit.edu
†SENSEable City Laboratory,Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA,
‡Institute of Mathematics, National Academy of Sciences of Belarus Republic, 11 Surganova str., Minsk, Belarus

2. redirecting links from each source node to all nodes, proportionally to the quota, at each destination, of the total in-weight of the network.

Modularity is then naturally bounded by $[-1, 1]$; a slightly more refined upper bound is given by summing only over the positive elements of the modularity matrix Q_{ij}.

1.2 Description Length

In this approach [2], one evaluates the fitness of a given partition of nodes to describe infinitely long random walks happening on the network. The fitness is quantified as the maximum compression one can achieve by assigning a hierarchical structure of code to communities, and considering how frequently each node will be visited.

Mathematically, the objective function is the average number of bits per step that is required to describe an infinite random walk on a network upon which a partition M is imposed:

$$L(M) = q_{out}\mathcal{H}^{out} + \sum_{C \in M} p_{in}^{C}\mathcal{H}_{C}^{in}; \tag{3}$$

the first term of the RHS gives the average description length for movement between different modules, the second term for movement within modules. In particular, q_{out} is the asymptotic probability of exiting from the current community, $\mathcal{H}^{out}$ the entropy of inter-module movement, p_{in}^{C} the asymptotic probability of remaining in community C, and $\mathcal{H}_{C}^{in}$ the corresponding entropy.

For a complete description, the reader is directed to Ref. [2].

2 Benchmark Networks

For our benchmark, we have a wide selection of networks, detailedly reported in Tab.2, which are divided into three groups:

- Networks 1-10 and 16-18 were previously used in papers ranging from biology to psychology, from human mobility to network science; they are all freely available. Relevant citations can be found alongside their description;
- Networks 11-15 result from telecom data we possess; the sources are under an NDA, and will thus remain private;
- Networks 19-23 are artificial structures with built-in communities; we obtained them using Lancichinecchi-Fortunato-Radicchi's algorithm [3], which is freely available at Fortunato's website[1]. The networks were created with average degree 8, maximum degree 16, mixing parameter 0.1, minimum and maximum community sizes 5 and 50, and β 1.

[1] https://sites.google.com/site/santofortunato/inthepress2

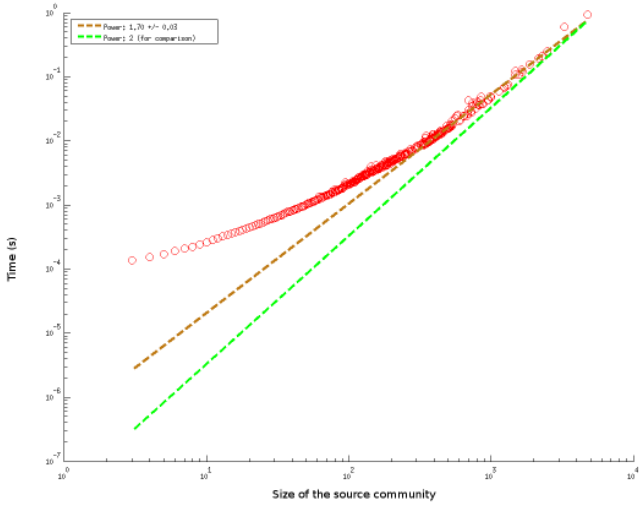

Figure 1: Variation of the execution time of a Kernigan's shift with the size of the source community.

3 Complexity Analysis for Combo

As Combo performs iterative optimizations at each step, its computational complexity cannot be sharply computed. Furthermore, the number of operations performed depends on the specific optimization allowed by the objective function used: in the following we'll discuss Combo for modularity, and denote by N the number of nodes in the network, by S and D the number of nodes in the source and destination communities currently considered, and by c the number of communities at a given iteration of the main Combo loop.

The fundamental unit of Combo is Kernigan's shift, in which all the nodes from a source community are sequentially switched to a destination community, with the best moves performed first. The computational complexity of each Kernigan's shift scales as the square of the number of nodes in the source community, although in actual computations a sizable overhead is present (see Fig.1).

Kernigan's shifts are iterated until no further improvement can be achieved; while the number of iterations cannot be anticipated, experimental observations show that its dependence on the size of the source community is present but weak, as shown in Fig.2.

Computing the best split between source and destination communities requires the calculation of a vector of weights (which account for the destination community) and the iteration of Kernigan's shifts until no gain is possible. The former step requires $\mathcal{O}(NS)$ operations, the latter $\mathcal{O}\left(S^2k\right)$, where k is the number of iterations of the Kernigan's shift; as the number of communities,

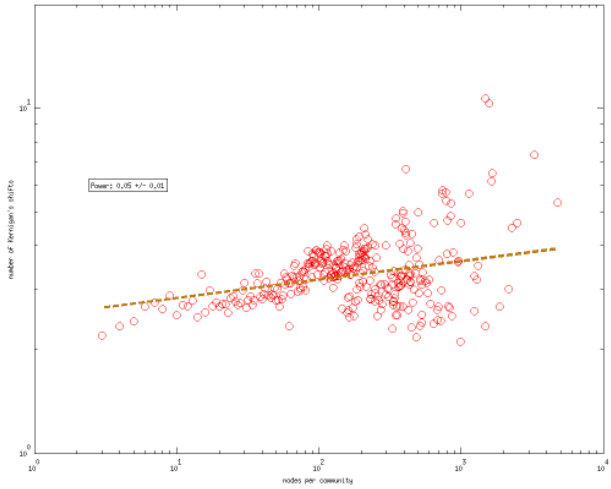

Figure 2: Variation with community size of the execution time of a Kernigan's shift. A power law with a low exponent is shown for comparison purposes.

on average, increases with network size faster than the iterations of Kernigan's shift, the computation of weight vectors would asymptotically dominate split operations. Profiling actually revealed that the vast majority of computation time is spent in Kernigan's shifts, probably because of fixed cost and function call overhead, thus we'll consider each split as $\mathcal{O}\left(S^2 k\right)$ for the range of explored network sizes.

In its main loop, Combo first selects the best possible split of a source into a destination community, then updates all the modularity gains. The latter operation requires the computation of about four times as many splits as the current number of communities; the exact cost of each operation depends on the size of the source community involved.

To a first approximation, we consider that the average number of nodes in the source community scales as N/c, where c is the number of communities at the current iteration, and a straightforward analysis of Combo's behaviour shows that the number of iterations of the main loop is roughly linear in the final number of communities (see Fig.3)

This entails that the computation time of each main loop scales as N^2/c; experimental observations show that the number of communities approximately increases at each loop until it gets very close to the final value, then slowly converges to the final result with almost no change in the number of communities. Keeping into account that the fraction of loops at which no change in the number of communities happens is approximately constant as the final number of communities $\mathcal{C}$ varies, the two phases take respectively $\mathcal{O}\left(N^2 \log\left(\mathcal{C}\right)\right)$ and $\mathcal{O}\left(N^2\right)$, thus Combo as a whole scales at worst as $N^2 \log \mathcal{C}$.

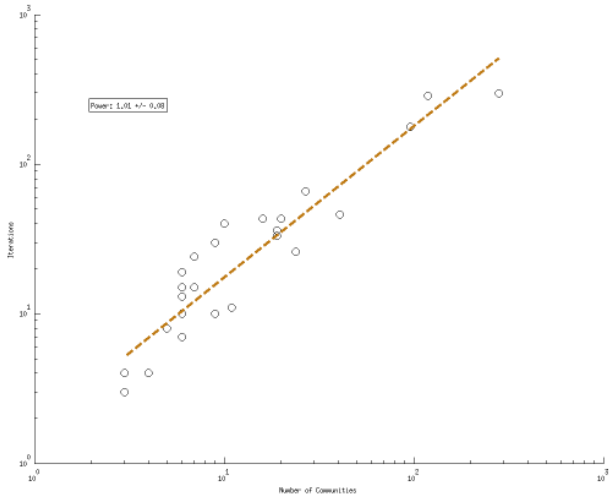

Figure 3: The number of iterations of the main loop is roughly linear in the final number of communities; their ratio varies from 1 to about 5.

This hypothesis is compatible with experimental data, which show that execution times increase more slowly than $N^2 \log \mathcal{C}$, while they are well described by a N^2 law, as shown, respectively, at the top and bottom of Fig.4.

4 Objective Function Optimization Results

The complete list of results from modularity and codelength benchmarks are reported in Tabs.3 and 5, respectively.

A summary of the modularity benchmark is reported in Tab.4. To obtain it, we first ranked the results of each algorithm based on modularity scores; when multiple algorithms achieved the same results, we attributed to each the best possible rank (e.g., if the third and fourth best modularities were the same, we would rank each algorithm as 3). Next, we normalized the rank on a $[0, 1]$ scale, with 1 corresponding to the best rank, 0 to the worst. Finally, for each algorithm we computed average and standard deviation of the normalized rank.

It's worthwhile to explicitly note that the second and third best algorithms, Le Martelot and Extremal Optimization, only work on undirected networks.

The execution times for each pair of network and algorithm is reported Fig.5.

5 LFR Benchmark - Description

Comparing the computed communities to the underlying structure of a network is the best way to evaluate the performance of different algorithms. However,

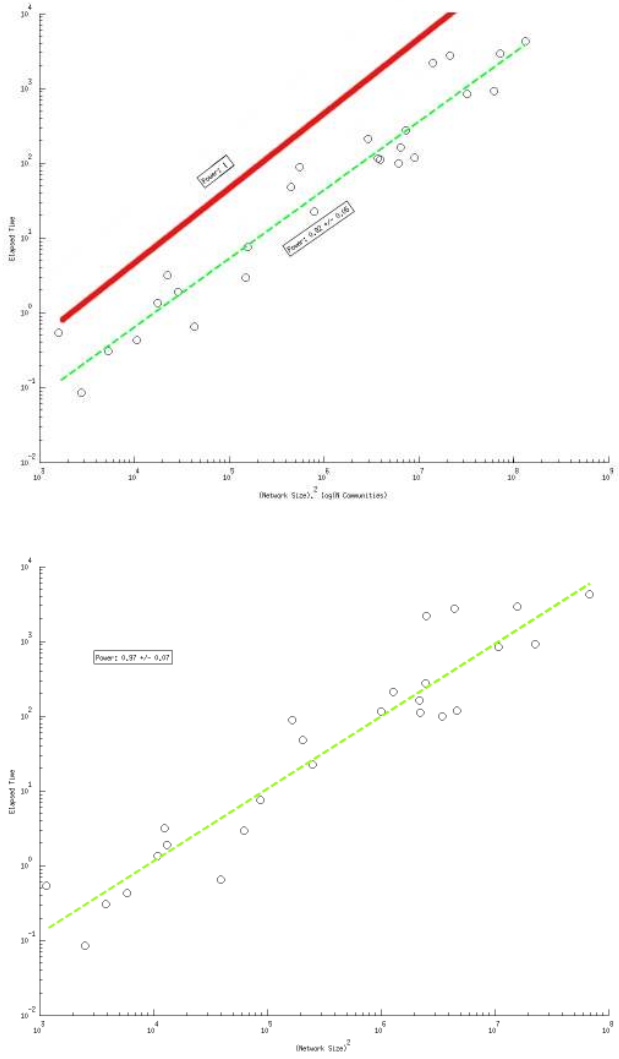

Figure 4: **Top** Elapsed time is shown together with $N^2 \log(\mathcal{C})$; the red line shows how the data would line up if it behaved according to the previous formula. **Bottom** Elapsed time is shown together with N^2. Green lines are the results of fitting a power law to the experimental data.

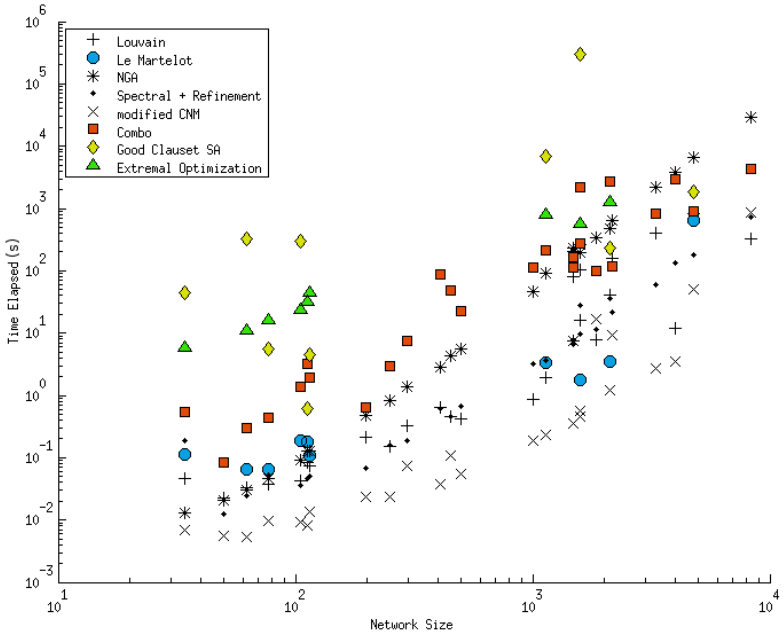

Figure 5: Execution times by network size and algorithm.

since there are few examples of such real-world networks in literature, the scientific community has mostly had to resort to artificially generated structures.

Several algorithms have been proposed for their creation; they mostly rely on glueing together densely inter-connected cliques.

One of the most popular of these methods - proposed by Girvan and Newman [4] - constructs simple networks made of equally sized communities with constant in- and outdegrees.

However, Girvan-Newman and other methods don't take into account some of the key properties of real-world networks, such as power-law distributions for vertex degrees and community size.

In 2008, Lancichinetti, Fortunato, and Radicchi proposed a method which overcomes these limitations [3], later extending it to cover weighted and directed networks [5]; since their benchmark has become increasingly popular in recent years, we decided to adopt it for the present work.

The main characteristics of the Lancichinetti-Fortunato-Radicchi method are: 1) vertex degrees and community sizes are chosen from power law distributions; 2) the number of links connecting different communities (outdegree) is a fixed fraction of the total number of links; 3) the same is true for link weights.

More specifically, in the implementation of the algorithm proposed by the authors, the in-degree sequence y_i is sampled from a power law, and the outdegree sequence z_i from a δ-distribution. Community sizes $\{S_\xi\}$ are also sampled from a power law. Afterwards, vertices are assigned to communities satisfying

$$\left.\begin{aligned}(S_\xi)_{i\in\xi} &\geqslant y_i^{(in)}\\ (S_\xi)_{i\in\xi} &\geqslant z_i^{(in)}\end{aligned}\right\} \quad \forall i$$

Each community is generated as a separate subgraph, in which multiple links are eliminated by rewiring. Then external links are added so that $y_i^{(ext)} = y_i - y_i^{(in)} = \mu_t\, y_i$ and $z_i^{(ext)} = z_i - z_i^{(in)} = \mu_t\, z_i$ as the **topological mixing parameter** μ_t is kept constant for in- and outdegree. At this stage we already have directed unweighted graphs with community structure and the desired distribution of vertex degrees and community sizes.

Next, the strength of each node is calculated as $s_i = (y_i + z_i)^\beta$. As pointed in [6], such a relation is frequently observed in real world networks. Internal and external strength are calculated using the **weight mixing parameter** μ_w: $s_i^{(in)} = (1-\mu_w)s_i$, $s_i^{(ext)} = \mu_w\, s_i$. To assign weights to links with respect to these strengths, the following steps are taken.

How close current weights are to the desired one is measured as

$$Var(\{w_{ij}\}) = \sum_i ((s_i - p_i)^2 + (s_i^{(in)} - p_i^{(in)})^2 + (s_i^{(ext)} - p_i^{(ext)})^2).$$

Here $p_i = \sum_j (w_{ij})$, $p_i^{(in)} = \sum_j w_{ij}\, C(i,j)$, $p_i^{(in)} = \sum_j w_{ij}(1 - C(i,j))$, and $C(i,j)$ indicates (i. e. $C(i,j) = 1$) that i and j belong to one community (and $C(i,j) = 0$ otherwise). A fast and simple greedy algorithm used to minimize it:

1. At the beginning all weights are set to zero, so $w_{ij} = 0, \forall i, j$, $p_i = 0$. Then, for all nodes i the next two steps are repeated:

2. Vertex i is chosen and all its link weights are increased by $\Delta w = \frac{s_i - p_i}{k_i}$. After that for each vertex i we have $p_i = s_i$, and we update values $\{p_i\}$.

3. For a given vertex i all the link weights w_{ij} are increased by an amount $\frac{s_i^{(in)} - p_i^{(in)}}{k_i^{(in)}}$ if $C(i,j) = 1$ and decreased by $\frac{s_i^{(in)} - p_i^{(in)}}{k_i^{(ext)}}$ if $C(i,j) = 0$ and $w_{ij} > \frac{s_i^{(in)} - p_i^{(in)}}{k_i^{(ext)}}$.

4. This process is repeated several times until a steady state or a certain value is reached.

6 LFR Benchmark - Results

To compare different partitions of the same network, we computed their Normalized Mutual Information (NMI) [7]: NMI measures the information-theoretic content of a pseudo-confusion matrix, whose entries N_{ij} are the number of nodes which are in community i for the first partition (A) and j for the second one (B). It is defined as

$$NMI = \frac{-2 \sum_{i=1}^{C_A} \sum_{j=1}^{C_B} N_{ij} \log\left(\frac{N_{ij} N}{N_i^A N_j^B}\right)}{\sum_{i=1}^{C_A} N_i^A \log\left(\frac{N_i^A}{N}\right) + \sum_{j=1}^{C_B} N_j^B \log\left(\frac{N_j^B}{N}\right)},$$

where C_A and C_B are the number of communities, N_i^A and N_j^B the cardinality of each community, N the number of vertices.

The comparison of extracted communities with the underlying structures in random networks was analyzed in the main text; detailed results are shown in Fig.6 and Fig.7.

7 Real Networks Benchmark

Networks with "known" community structure constitute a second type of benchmark. Such networks are however very limited in number. Most of the networks with known community structures were previously considered in the scientific literature, and bear precise information about vertices and their properties.

To compare different objective functions for the reconstruction of the community structure of real-world networks, we chose six networks previously considered in literature, whose underlying community structure is commonly agreed upon as known.

1. **(karate)** Network of friendship relations between members of a US university karate club, known in literature as Zachary karate club [8]. This

graph is well known and often used as a benchmark for community detection algorithms. The club consisted of 34 members and after internal disagreements it broke up in two groups.

2. **(football)** Network of American football games between Division IA colleges during regular season Fall 2000 [4]. There are 115 teams, corresponding to vertices, pairs of which are connected by an edge if they played each other. All teams are separated into 12 conferences. Conferences offer a natural community structure, as teams from one conference play more often one another than teams from a different conference.

3. **(UKfaculty)** The personal friendship network of the faculty of a UK university consisting of three separate schools [9]. The network consists of 81 vertices (individuals) and 817 directed and weighted connections. This dataset contained explicit information regarding the expected community structure, since we know which school each node belongs to (with the exception of two nodes, that do not belong to any).

4. **(macaque)** Graph model of the visuo-tactile brain areas and connections of the macaque monkey [10]. The graph consists of 45 vertices representing brain areas, and 463 directed connections representing neuronal pathways between the areas. Two distinct and mostly non-overlapping communities correspond to the visual and the somatosensory cortex.

5. **(polbooks)** A network of books on politics, compiled by V. Krebs (unpublished, see http://www.orgnet.com). In this network, the vertices represent 105 recent books on American politics sold by the online bookseller Amazon.com, and edges join pairs of books that have been purchased together by many users. Books were divided according to their stated or apparent political alignment – liberal or conservative – except for a small number of books that were explicitly bipartisan or centrist, or had no clear affiliation.

6. **(polblogs1222)** A network of political blogs assembled by Adamic and Glance [11]. The network is composed of blogs about US politics and the web links between them, as captured on a single day in 2005. The blogs have known political leanings and were labeled by Adamic and Glance as either liberal or conservative; directed edges connect vertices if one of the corresponding blogs contained a hyperlink to the other on its front page. We only considered the network's largest connected component, which has 1222 vertices.

8 The Effect of Small Changes of Objective Functions on Partitions

As discussed in the main text, small changes in the value of the objective function can be reflected by macroscopic variation of the communities involved.

Table 1: The partition sporting the best modularity score (incidentally for all six networks it is the one obtained by using Combo) is compared to a close runner up, chosen to show that even small variations can lead to substantial differences in the resulting community structure.

network	Mod Best	Mod Alt	NMI
1	0.57524	0.56723	0.93654
2	0.6058	0.6057	0.9345
3	0.4449	0.4345	0.8538
4	0.3840	0.3821	0.9255
5	0.4414	0.4337	0.5564
6	0.5272	0.5244	0.9166

To illustrate this point, for each of the six networks we used in the previous benchmark, we compared the partition with the highest modularity score with hand picked close runner-ups: as shown in Tab.1, very low differences in modularity can correspond to large variations of normalized mutual information.

9 Relaxed Caveman, I-Partition, and Gaussian Random Graph Benchmark

We also compared the relative effectiveness of modularity and code length optimization in three other types of graphs:

1. The class of relaxed caveman graphs. A relaxed caveman graph starts with separeted cliques of given size. Edges are then randomly rewired with probability p to link different cliques [24].

2. Graphs generated with the so-called planted l-partition model. The model partitions a graph with $n = gl$ vertices in l groups with g vertices each. Vertices of the same group are linked with a probability p_{in}, and vertices of different groups are linked with a probability p_{out} [24,25]. For our tests we set $p_{in} = 0.6$ and chose p_{out} as $p_{out} = \frac{scale*50}{n}$, where n is number of nodes in the graph.

3. Graphs generated via a Gaussian random partition generator – a modified version of the planted l-partition model where cluster sizes have a Gaussian distribution with given mean and variance [26]. We changed p_{out} in the same way as for the planted l-partition model.

For each benchmark we generated networks of sizes 250, 500 and 1000 nodes and communities of sizes 15, 20 and 25 nodes. We used the NetworkX library [27] to generate this graph.

Table 2: List, with sources, of the networks we used in our benchmark.

Network	**Nodes**	**Description**
1	34	Zachary's Karate network [8]
2	62	Dolphins' Social Network [12]
3	77	Coappeareance of characters in Les Miserable [13]
4	105	Amazon.com Co-purchases of political books [2]
5	112	Common adjective and noun adjacencies in David Copperfield [14]
6	115	American College Football games in year 2000 [15]
7	297	Neural network of C. Elegans [16]
8	1490	Connections among political blogs [17]
9	1589	Coauthorship in network science [14]
10	2114	Protein interaction network for Saccharomyces Cerevisiae [18]
11	2163	Portugal mobile phone network
12	4761	UK mobile phone network
13	3296	Portugal network from radiation model
14	1479	UK network from radiation model
15	1579	France network from radiation model
16	8297	Wiki vote network [19]
17	1858	Complete network of US airports in 2010 [3]
18	410	Network extracted from the Infectious: STAY AWAY exhibition [20]
19	50	Synthetic network of 50 nodes [3]
20	250	Synthetic network of 250 nodes
21	500	Synthetic network of 500 nodes
22	1000	Synthetic network of 1000 nodes
23	4000	Synthetic network of 4000 nodes
24	1133	Email Networks University of Tarragona [21]
25	198	Network of Jazz Musicians [22]
26	453	Metabolic Network of C. Elegans [23]

2. Valdis Krebs, unpublished
3. data from the Bureau of Transportation Statistics - details at `http://toreopsahl.com/datasets/#usairports`

Table 3: Performance comparison of Louvain method, Le Martelot algorithm, Newman's greedy algorithm (NGA), Newman's spectral method with refinement, Simulated Annealing, Extremal Optimization, and our new method (abbreviated as "Combo").

Network	Size	Louvain	Le Mar	NGA	Sp+Ref	G-C SA	Ext Opt	Combo
1	34	0.4188	0.4198	0.3807	0.4188	0.4198	0.4198	0.4198
2	62	0.5188	0.5233	0.4955	0.5265	0.5276	0.5265	0.5268
3	77	0.5654	0.5667	0.5472	0.5658	0.5656	0.5658	0.5667
4	105	0.4986	0.5268	0.5020	0.5244	0.5272	0.5272	0.5272
5	112	0.2906	0.2993	0.2947	0.2985	0.3028	0.3006	0.3051
6	115	0.6021	0.6053	0.5720	0.6018	0.6054	0.6054	0.6054
7	297	0.5048	0.3485	0.5155	0.5024			0.5178
8	1490	0.4311		0.4318	0.4316			0.4318
9	1589	0.9451	0.9546	0.9543	0.9467	0.9485	0.9550	0.9550
10	2114	0.7841	0.8401	0.8458	0.8142	0.8317	0.8442	0.8512
11	2163	0.4448		0.4689	0.4857			0.4888
12	4761	0.6312	0.6535	0.6528	0.6500	0.5753		0.6577
13	3296	0.8374		0.8669	0.8706			0.8761
14	1479	0.8253		0.8483	0.8527			0.8580
15	1579	0.8196		0.8448	0.8435			0.8520
16	8297	0.4214		0.3353	0.4271			0.4297
17	1858	0.2739		0.2542	0.2610			0.2756
18	410	0.8505		0.8607	0.8500			0.8610
19	50	0.6419		0.6419	0.6419			0.6419
20	250	0.8014		0.7760	0.8014			0.8014
21	500	0.8530		0.8318	0.8530			0.8530
22	1000	0.8756		0.8577	0.8752			0.8756
23	4000	0.8974		0.8821	0.8949			0.8974
24	1133	0.5406	0.5741	0.5036	0.5627	0.4852	0.5740	0.5825
25	198	0.4349		0.4386	0.4370			0.4454
26	453	0.4429		0.4237	0.4336			0.4522

Table 4: Summary of the results. To each algorithm we associated its average rank (normalized to the interval $(0, 1)$, where 0 is the worst result, 1 the best) in the 26 (10 in the case of Le Martelot and Extremal Optimization) networks it has been run on, and the corresponding standard deviation.

Algorithm	avg score	std dev
Louvain	0.371	0.356
Le Martelot	0.593	0.248
Newman's greedy	0.377	0.333
Spectral + Refinement	0.475	0.219
Combo	0.957	0.092
Good-Clauset SA	0.529	0.378
Extremal Optimization	0.762	0.117

Table 5: Optimal compression length achieved by Infomap and Combo. Lower scores correspond to better compression.

Network	Size	Infomap	Combo
1	34	4.6061	4.6061
2	62	5.302	5.3026
3	77	4.8384	4.8384
4	105	5.923	5.923
5	112	6.481	6.481
6	115	5.9442	5.9442
7	297	7.0105	7.0887
8	1490	9.1169	9.0380
9	1589	4.6877	4.6861
10	2114	6.0164	6.111
11	2163	10.438	10.2478
12	4761	10.611	10.0253
13	3296	7.7975	7.8340
14	1479	7.0998	7.0914
15	1579	7.1834	7.1786
16	8297	11.752	11.9133
17	1858	7.827	7.7909
18	410	6.7632	6.7668
19	50	4.7611	4.7611
20	250	6.034	6.0340
21	500	6.1011	6.1011
22	1000	6.3542	6.3542
23	4000	6.9161	6.9763
24	1133	8.6073	8.6076
25	198	6.717	6.7158
26	453	7.5107	7.7039

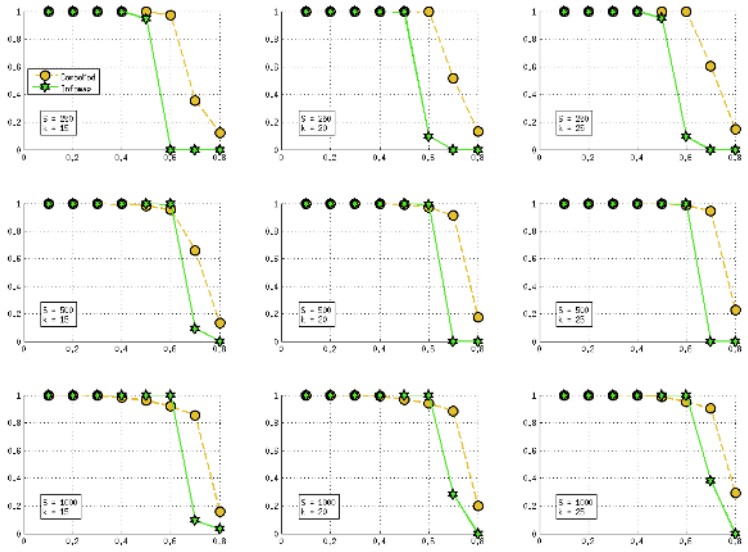

Figure 6: We present here a comparison between optimization of modularity and code length. The x coordinate represents the mixing factor μ_w; the y coordinate is normalized the mutual information. The topological mixing factor μ_t is set to 0.5.

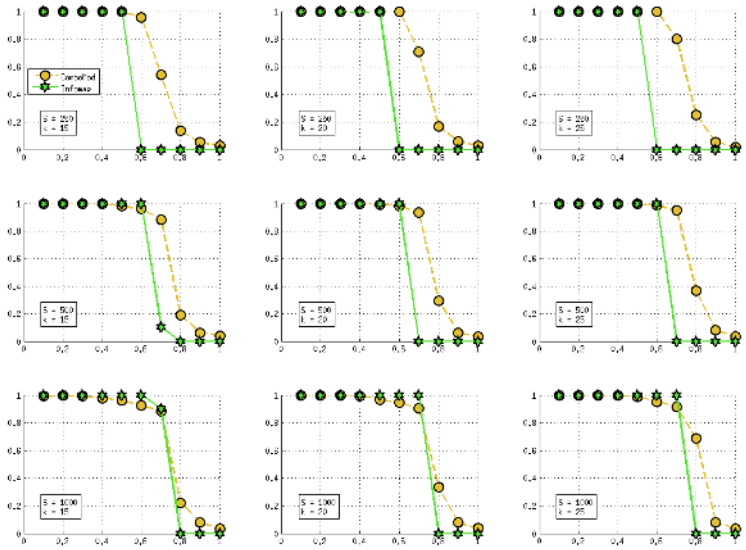

Figure 7: The topological mixing factor μ_t is equal to μ_w.

Results from this benchmarks are reported in Fig.8: as in the results of the LFR benchmark, modularity yields more reliable community reconstruction as the level of noise increases; code length performs surprisingly poorly for smaller networks.

References

[1] A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev.*, E70 (6):066111, 2004.

[2] M. Rosvall and C.T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proc. Natl. Acad. Sci. USA*, 105:1118–1123, 2008.

[3] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78 (4):046110, 2008.

[4] M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99 (12):7821–7826, 2002.

[5] A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E.*, 80(1):016118, 2009.

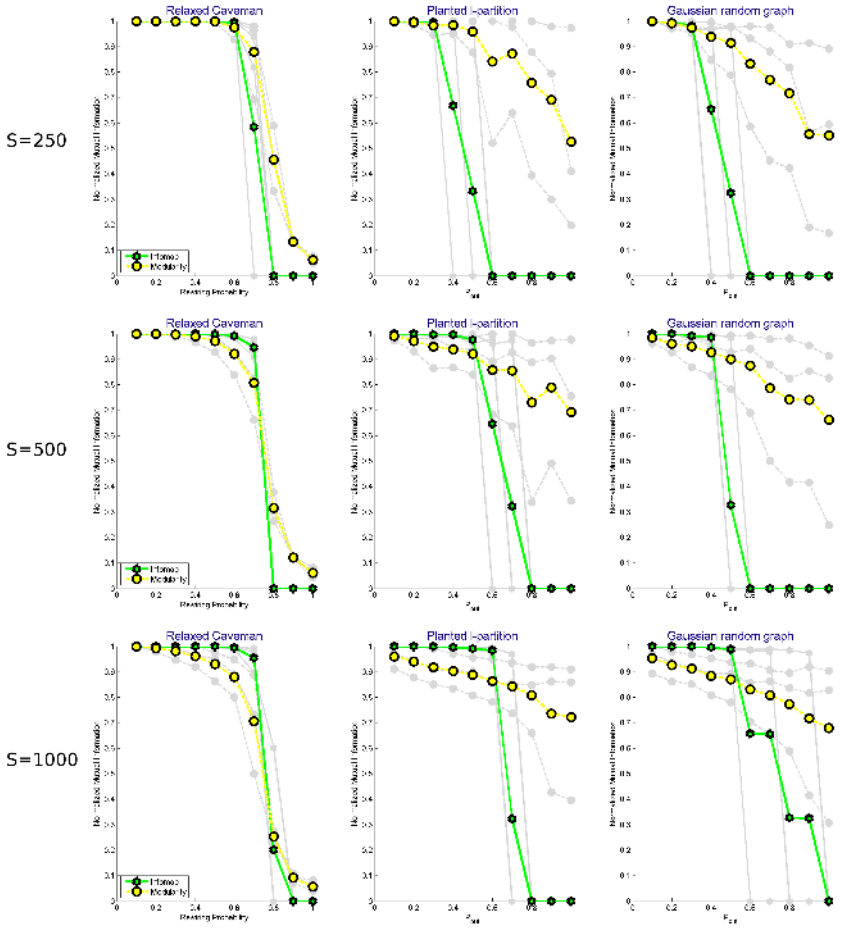

Figure 8: Relaxed Caveman, I-Partition and Gaussian Random Graph benchmarks. Colored markers result from the average of shaded ones. Modularity was computed using Combo.

[6] A. Barrat, M. Barthlemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(11):3747–3752, 2004.

[7] Leon Danon, Albert Daz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, 2005.

[8] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

[9] T. Nepusz, A. Petroczi, L. Negyessy, and F. Bazso. Fuzzy communities and the concept of bridgeness in complex networks. *Phys. Rev. E*, 77:016107, 2008.

[10] L. Negyessy, T. Nepusz, L. Kocsis, and F. Bazso. Prediction of the main cortical areas and connections involved in the tactile function of the visual cortex by network analysis. *European Journal of Neuroscience*, 23(7):1919–1930, 2006.

[11] L.A. Adamic and N. Glance. The political blogosphere and the 2004 US Election. *Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem*, 2005.

[12] David Lusseau, Karsten Schneider, Oliver J. Boisseau, Patti Haase, Elisabeth Slooten, and Steve M. Dawson. The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

[13] D. E. Knuth. *The Stanford GraphBase: a platform for combinatorial computing*. Addison-Wesley, 1993.

[14] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74:036104, Sep 2006.

[15] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[16] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner. The structure of the nervous system of the nematode caenorhabditis elegans. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 314(1165):1–340, 1986.

[17] Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 u.s. election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, LinkKDD '05, pages 36–43, New York, NY, USA, 2005. ACM.

[18] H. Jeong, S. P. Mason, A.-L. Barabasi, and Z. N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, May 2001.

[19] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1361–1370, New York, NY, USA, 2010. ACM.

[20] Lorenzo Isella, Juliette Stehl, Alain Barrat, Ciro Cattuto, Jean-Franois Pinton, and Wouter Van den Broeck. What's in a crowd? analysis of face-to-face behavioral networks. *Journal of Theoretical Biology*, 271(1):166 – 180, 2011.

[21] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Phys. Rev. E*, 68:065103, Dec 2003.

[22] Pablo M. Gleiser and Leon Danon. Community structure in jazz. *Advances in Complex Systems*, 06(04):565–573, 2003.

[23] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Phys. Rev. E*, 72:027104, Aug 2005.

[24] S. Fortunato. Community detection in graphs. *Physics Report*, 486:75–174, 2010.

[25] Anne Condon and Richard M Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2):116–140, 2001.

[26] Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. *Experiments on graph clustering algorithms*. Springer, 2003.

[27] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008.